Taller de Tastypie y AngularJS

Python Valencia, 7/5/2014

APIs RESTful

Representational State Transfer (REST)

Conjunto de **principios de arquitectura** con los que diseñar servicios Web enfocados a los **recursos** del sistema.

Principios REST

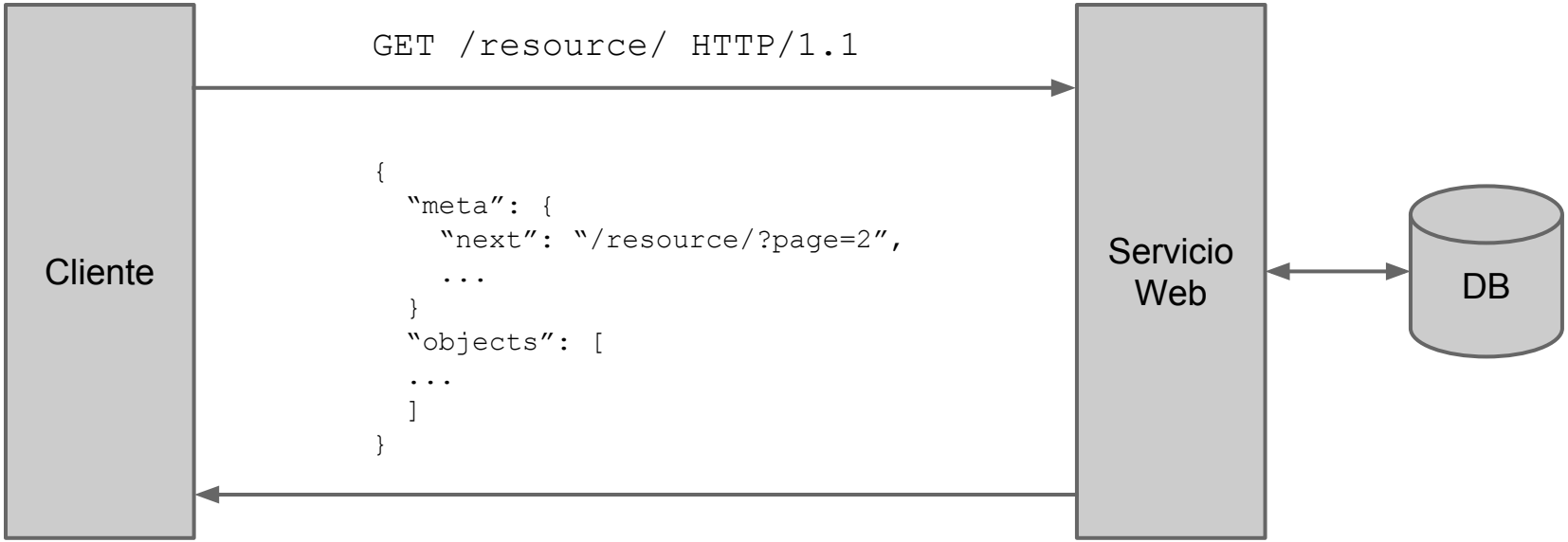
- Uso de métodos HTTP de forma explícita
- Sin estado
- Estructura de URIs basada en directorios
- Transferencia de datos en JSON, XML o ambos

Uso de métodos HTTP explícitamente

Operaciones CRUD:

- POST para **crear** un recurso
- GET para **obtener** un recurso
- PUT/PATCH para **actualizar** un recurso
- DELETE **borrar** un recurso

Diseño sin estado



Estructura de URIs basada en directorios

Las URIs de un servicio web REST tienen que ser **intuitivas**.

```
/noticias/
/noticias/(:id)/
/noticias/(:id)/comentarios/
```

Transferencia de datos en JSON/XML

```
GET /api/v1/characters/1/ HTTP/1.1
Content-Type: application/json
{
   "base_dexterity":12,
   "base_intelligence":11,
   "base_strength":16,
   "dexterity":1,
   "id":2,
   "intelligence":0,
   "name":"Erik",
   "profession":{
      "id":2,
      "name":"Warrior",
      "resource_uri":"/api/v1/professions/2/",
      "skills":[]
   },
   "race":{
      "dexterity_modifier":1,
      "id":2,
      "intelligence_modifier":1,
      "name":"Human",
      "resource_uri":"/api/v1/races/2/",
      "strength_modifier":0
   },
   "resource_uri":"/api/v1/characters/2/",
   "strength":3
}
```

Implementando un API RESTful

¿Que vamos a usar?

- **Python (por supuesto)**
- **Django - https://www.djangoproject.com/**
- **Tastypie - http://tastypieapi.org/**

Código fuente

```
$ git clone git@github.com:
pythonvlc/workshop-tastypie-
angularjs.git
```

Instalación

```
$ cd workshop-tastypie-angularjs
$ mkvirtualenv workshop
(workshop)$ pip install -r
requirements/local.txt
```

Comprobar que todo funciona

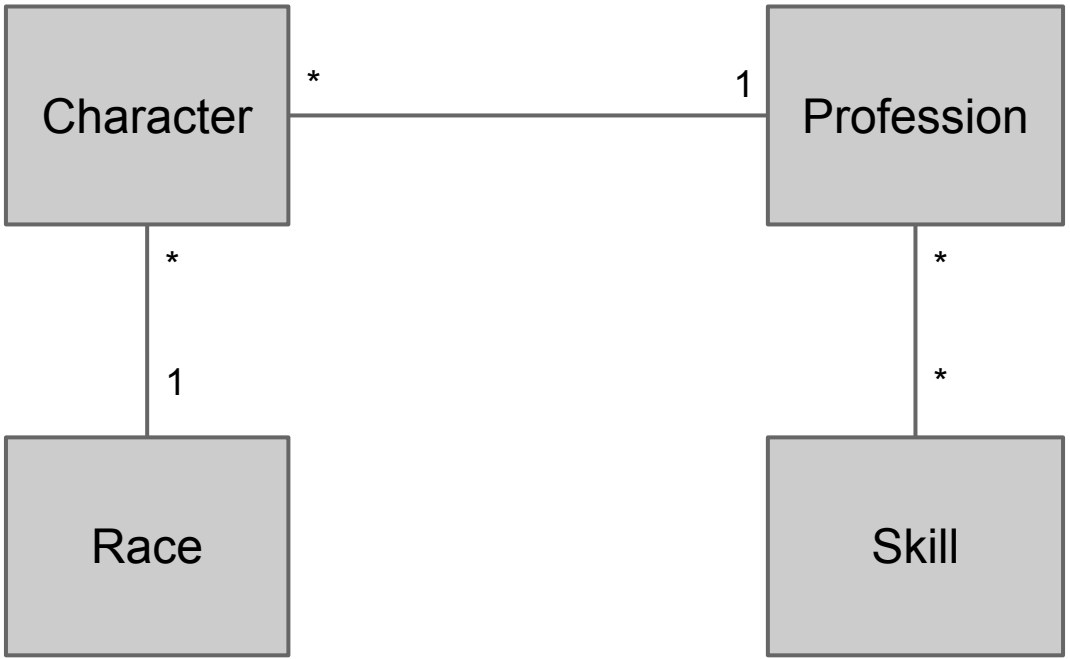
```
(workshop)$ cd simplerolegame/
(workshop)$./manage.py test
Creating test database for alias 'default'...
..........
----------------------------------------------------------------------
Ran 10 tests in 0.071s

OK
Destroying test database for alias 'default'...
```

Definición del problema

Implementar un API para **juegos de rol minimalistas y genéricos**.

- **Personajes**, tienen atributos (fuerza, destreza, inteligencia)
- **Raza**, modifica los atributos
- **Profesión**, define qué habilidades tiene el personaje
- **Habilidades**, lo que puede hacer el personaje

Modelos



Tastypie

- Clases `Resource` y `ModelResource`
  - `Resource` para colecciones de datos de cualquier tipo
  - `ModelResource` para usar directamente con modelos de Django.

Tastypie

```
class RaceResource(ModelResource):

    class Meta:
        object_class = Race
        queryset = Race.objects.all()
        resource_name = "races"
        authorization = Authorization()
        authentication = Authentication()
```

Tastypie

```
class ProfessionResource(ModelResource):

    skills = fields.ToManyField(SkillResource, attribute='skills', full=True, null=True)

    class Meta:
        object_class = Profession
        queryset = Profession.objects.all()
        resource_name = "professions"
        authorization = Authorization()
        authentication = Authentication()
```

Tastypie

```
# urls.py
api_v1 = Api(api_name='v1')
api_v1.register(CharacterResource())
api_v1.register(RaceResource())
api_v1.register(ProfessionResource())
api_v1.register(SkillResource())
urlpatterns = patterns(
    '',
    url(r'^api/', include(api_v1.urls)),
)
```

Nuevas *features*

1. Crear un nuevo recurso *weapons*, que otorgará habilidades extras a los personajes.

2. Crear un inventario para los *characters*, donde puedan añadir y quitar *weapons*.

¡Gracias!